Fabric defect detection using the wavelet transform in an ARM processor

J. A. Fernández [a], S. A. Orjuela [b], J. Álvarez [a] and W. Philips [b]

[a] Nariño University, Ibagué, Colombia;
[b] Department of Telecommunications and Information Processing (TELIN), IPI, IBBT, Gent University, Gent, Belgium;

ABSTRACT

Small devices used in our day life are constructed with powerful architectures that can be used for industrial applications when requiring portability and communication facilities. We present in this paper an example of the use of an embedded system, the Zeus epic 520 single board computer, for defect detection in textiles using image processing. We implement the Haar wavelet transform using the embedded visual C++ 4.0 compiler for Windows CE 5. The algorithm was tested for defect detection using images of fabrics with five types of defects. An average of 95% in terms of correct defect detection was obtained, achieving a similar performance than using processors with float point arithmetic calculations.

Keywords: Experimental designs, image analysis, texture analysis

1. INTRODUCTION

Nowadays, advances in technology offers portable devices for our day life with entertainment and communication applications as well as customizable capabilities. The system architectures of these devices are a powerful tool to develop friendly visual environments for industrial applications such as inspection, control or monitoring tasks. Furthermore, these systems operate in reduced working spaces and offer portability capabilities that are useful for visual inspection tasks among others. The software for portable devices can be developed by using a virtual machine, which facilitates the implementation of applications by being independent of the instructions set of the processor. Software based on a virtual machine, i. e. Android, are ideal for multimedia applications.

Processing speed and portability are fundamental aspects when developing industrial tools. Therefore, despite the advantages of the virtual machine in facilitating data manipulation, for real time applications it is preferred to have direct access to the instructions set of the processor to ensure a fast data processing. Platforms for developing embedded portable systems are currently programmed using C++, which facilitates data manipulation by optimizing structures. Portable devices for real time applications can then be developed using platforms like Windows CE and Embedded Linux. Embedded systems pose a limited architecture and the right choice of the hardware depends on the application requirements. The most dominant architectures for embedded electronics market are based on Acorn RISC Machine (ARM) processors, which are appropriate for low power applications due to their relatively simplicity. Applications of ARM processors include consumer electronics products such as mobile phones, tablets and game consoles as well as computer peripherals such as hard drives and routers.

Further author information: (Send correspondence to J. A. Fernndez)
Antonio Nariño University, Cra. 10 # 17 - 35, Ibagué - Colombia: E-mail: fernandezgallego@gmail.com, Telephone: +57 311 2104189, +57 8 2612003

We present in this paper an example of the use of an embedded system, the Zeus Epic 520 device, for image processing applications by implementing the wavelet transform using the Embbeded Visual C++ 4.0 compiler for Windows CE5. In one hand, the ZEUS EPIC 520 device developed by Eurotech has a ARMV4I processor. It also has a tactile screen, an integrated camera and a video output (VGA), which makes it suitable for visual inspecting tasks. Furthermore, it has communication capabilities such as Ethernet, RS 232, RS 422, RS 485, USB 1.1 and a Wi-Fi module. SD, MMC or USB memories can be used for saving data. In the other hand, the Wavelet transform is defined as as multi-resolution analysis of a finite energy function.[1] Wavelet transform offers a compromise between spatial and frequency localization of the image features. It shows good performance for several tasks of texture including recognition, classification and segmentation.[2–11] Research on implementing the wavelet transform in embedded systems has been conducted using DSP and FPGA for images,[12] and in ARM processors mostly for unidimensional signals.[13]

The wavelet multi-resolution analysis of an image performs a filter bank decomposition of the image using low pass and high pass filters.[1,9] The decomposition of an image can be performed by decomposing their rows and columns as unidimensional signals.[1] This permits to use the basic operations of the ARM processor by implementing a series of addition and multiplication operations on the rows and columns of the image. We implemented the pyramidal wavelet decomposition, which decompose the image by sequentially analysing rows and columns.[1,14–16] Several types of mother wavelet, including Haar, Daubechies, Symlet, Biortogonal, Reverse biorthogonal, Coiflet and Discrete meyer among others, can be used in the pyramidal algorithm to decompose an image.[5,17,18] Among them, the Haar wavelet has proved an acceptable performance in applications of texture analysis.[6,7,9] Therefore, we use the Haar wavelet in this approach. Despite the ARM processor use fixed point arithmetic calculations, we were able to achieve similar results like when using float point arithmetic calculation. The algorithm was tested for defect detection using images of fabrics with five types of defects.

Texture features of the images from the resulting wavelet decomposition are extracted using a set of Haralick descriptors that have been found appropriate for texture classification.[19,20] To recognize images exhibiting texture defects we use a Mahalanobis classifier.[21] The paper is organized as follows. In Section 2 we discuss how the Wavelet Haar was implemented in the ARM processor. In Section 3 we discuss the method for extracting features using Haralick descriptors. In Section 4 we report the results and discuss the findings. Finally, in Sections 5 conclusions are drawn.

2. IMPLEMENTING THE WAVELET TRANSFORM IN THE ARM PROCESSOR

The implemented pyramidal wavelet decomposition algorithm can process images with rectangular size up to $N \times M = 640000$ pixels, with N and M the number of rows and columns in the image. This pyramidal decomposition is based on Quadrature Mirror Filters (QMF). A QMF split one image into two subbands, which are images, each with half size of the original. The decomposition starts by applying a low and a high pass filters on the original image. Then, both filters are again applied on the resulting images to obtain four images in each decomposition level. The process is repeated on the double low pass filtered image to obtain the following four images of the next decomposition level. Therefore, each decomposition level is represented with three images except the last level which has an extra image, termed the approximation image. The scheme of the implemented pyramidal Wavelet transform is illustrated in Figure 1. Figures 2a and 2b show similar results of the pyramidal algorithm respectively implemented in an intel and an ARM processor.

Let X_0 be an image with size $N \times M$, with n and m the row and column positions of the pixel. In the following, we discuss how to implement the pyramidal wavelet deconvolution of X_0 by using low and high pass filters, denoted by $\tilde{h}$ and $\tilde{g}$, with $\tilde{h}$ and $\tilde{g}$ a pair of kernel vectors containing the coefficients of the mother Wavelet e.g. $\tilde{h} = [0.7071 \quad 0.7071]$ and $\tilde{g} = [-0.7071 \quad 0.7071]$ for the Haar Wavelet, and $\tilde{h} = [0.0352 \ -0.0854 \ -0.1350 \ 0.4599 \ 0.8069 \ 0.3327]$ and $\tilde{g} = [-0.3327 \ 0.8069 \ -0.4599 \ -0.1350 \ 0.0854 \ 0.0352]$ for the Daubechies Wavelet.

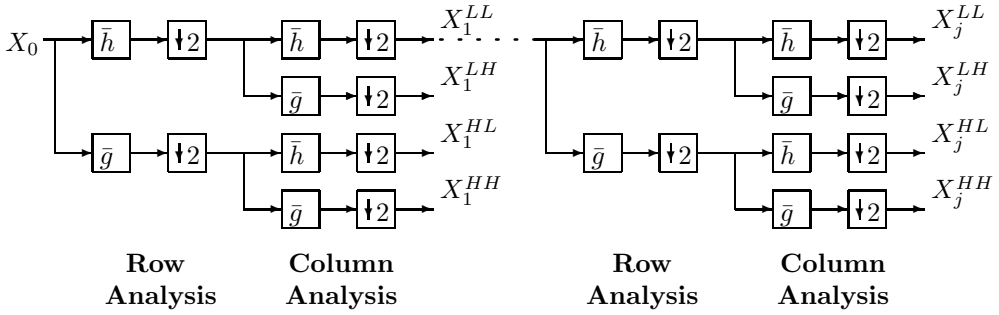
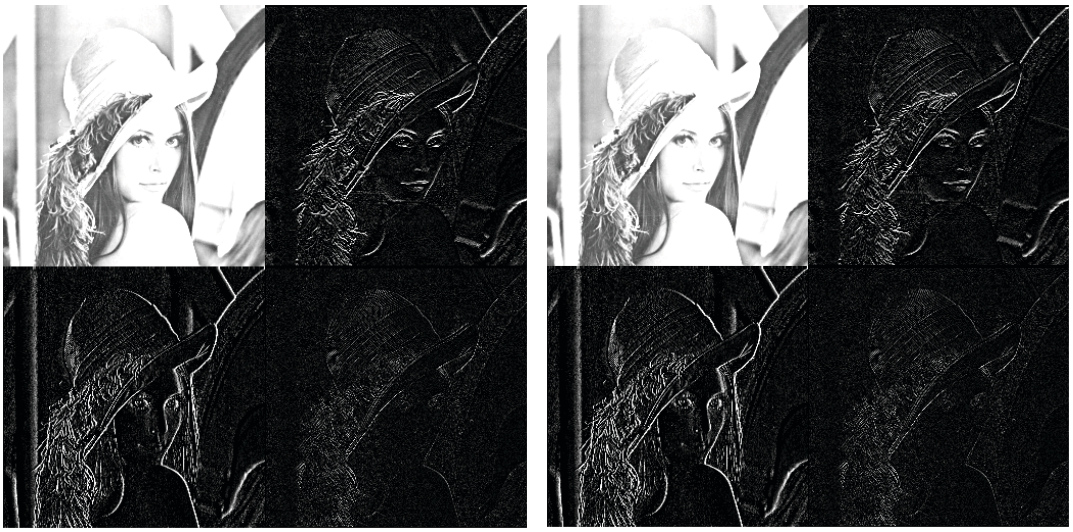


Figure 1: Decomposition process implemented in the ARM processor. X represents the input image. $\tilde{h}$ and $\tilde{g}$ are respectively the low and high pass filters. $\downarrow 2$ refers to the process of sub-sampling an image by a factor of two. X_j^{LL}, X_j^{LH}, X_j^{HL} and X_j^{HH} are the resulting images at each level of decomposition j. X_j^{LL} is the approximation image, in which the next level of decomposition is performed. The other images contain the horizontal, vertical and diagonal information of the image.[18]



(a) (b)

Figure 2: Results of applying the Haar wavelet transform in a classical test image (Lena). Figure (a) was obtained using the wavelet toolbox of Matlab. Figure (b) was obtained using the ARMv 4I processor.

We discuss in the following only the implementation of the first level of decomposition using f_1 and f_2 respectively as the first and second filters, which can be either $\tilde{h}$ or $\tilde{g}$. The images of the decomposition are obtained using the scheme in Figure 3 by replacing $\tilde{h}$ or $\tilde{g}$ by f_1 or f_2. Similarly, A and B refer to either Low L or High H pass filters. The intermediate images in the decomposition process are identified with the sub-scripts d.

By transposing both, the intermediate image X_d^A and the output image X_1^{AB}, the scheme in Figure 3 is performed by applying twice the convolution and down-sampling processes on the rows of the images X_0 and $(X_d^A)^T$, with the symbol $(.)^T$ denoting the transpose of an image. Therefore, by representing the pixels of a row n, of any of both images, by the vector $x_n(i)$, the basic process to implement in the ARM processor is shown in Equation 1, with f a kernel filter and $y_n(i)$ the output signal.

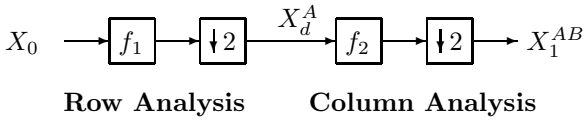
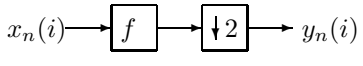


Figure 3: X_0 input image; ${X_d}^A$ intermediate image; (X_1^{AB}) image at the first level of decomposition.



$$y_n(i) = \sum_k x_n(k)\, f(2i-k) \tag{1}$$

With Equation 1, the computation of the subband images X_1^{AB} of Figure 3 are calculated by using the algorithm shown in Figure 4, which is explained in the following.

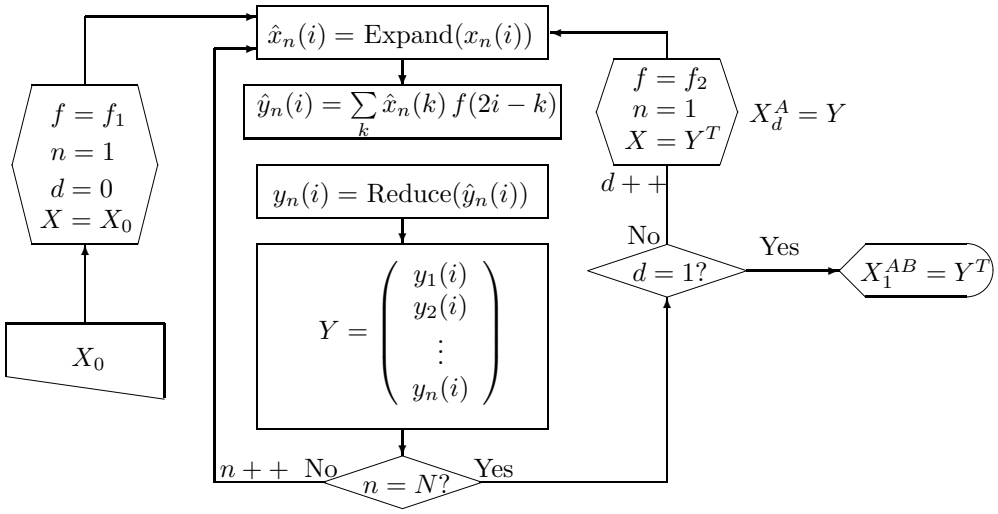


Figure 4: Flow diagram of the algorithm implemented in the ARM processor.

$y_n(i)$ in Equation 1 is implemented using neighbourhood operations. Because the vector $x_n(i)$ do not have neighbours at the borders, $(K-1)/2$ extra elements are commonly placed at each side of the vector to deal with the kernel computation. The extra elements are calculated using zero surround, nearest neighbour, extrapolation or circular cyclic methods among others.[22] For computational ease we use the nearest neighbour method. We denote the expanded vector of $x_n(i)$ by $\hat{x}_n(i)$. An extended vector $\hat{y}_n(i)$ is then computed by replacing $\hat{x}_n(i)$ by $x_n(i)$ in Equation 1.

The split images from a QMF may be with half size of the original. Therefore, extra elements of $\hat{y}_n(i)$ are eliminated to afterwards obtain a vector $y_n(i)$ with half size of $x_n(i)$. Finally, each processed image of Figure 3 is constructed by concatenating the resulting $y_n(i)$ vectors, when orderly processing all rows. We compute $x_n(i) = 1000x_n(i)$ and $y_n(i) = y(i)_n/1000$ to deal with the fixed point arithmetic calculations of the ARM processor. The full process is applied on X_0 using f_1 and then in $(X_d^A)^T$ using f_2. The decomposed image is obtained by transposing the last Y result.

3. FEATURE EXTRACTION FROM THE DECOMPOSED IMAGES

To recognize defective fabrics, we characterize the texture on the decomposed images computing features by applying the co–ocurrence matrix technique. The co–occurrence matrix is a square matrix, of order equal to the number of distinctive gray-level pixel values in the image, representing the distribution of co-occurring neighbour pixel values in an image.[19] The position (i, j) in the co–ocurrence matrix represents the number of times that any pair of disjoint pixels in the image, separated by a given distance d and an angle θ, have gray level pixel values i and j.

Rotational invariance is achieved by first computing few co–ocurrence matrices, for the same distance and different angles, and then adding them. Commonly, displacement vectors with the nearest neighbour at orientations $0°, 45°, 90°$ are used.[23–25] We denote the resulting combined co–ocurrence matrix of an image X as $\text{Cooc}(X)$. The original work proposed to characterize texture in an image by computing 14 features, called the Haralick descriptors, from a co-ocurrence matrix.[26] In this work we use the five features (discussed below) that are more commonly used.[20]

Let N_g be the number of distinct gray levels in the image and $p(i, j)$ the $(i, j)th$ entry in a normalized co–ocurrence matrix, where the sum of all its elements is equal to one. Consequently, $p_x(i)$, is the entry obtained by adding the values in the rows of $\text{Cooc}(X)$ and $p_y(j)$ is the entry obtained by adding the values in the columns of $\text{Cooc}(X)$. With these notations, the five features are computed as follows:

$$r_1 = \sum_i \sum_j (p(i,j))^2, \; Energy \tag{2}$$

$$r_2 = \sum_{q=0}^{N_g-1} q^2 \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \right), q = |i-j| \; Contrast \tag{3}$$

$$r_3 = \frac{\sum_i \sum_j (i,j) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}, \; Correlation \tag{4}$$

where μ_x, μ_y, σ_x, σ_y are the means and standard deviations of p_x and p_y.

$$r_4 = \sum_i \sum_j \frac{1}{1+|i-j|} p(i,j), \; Homogeneity \tag{5}$$

$$r_5 = -\sum_i \sum_j p(i,j) log(p(i,j)), \; Entropy \tag{6}$$

4. RESULTS: RECOGNIZING DEFECTIVE FABRIC SAMPLES

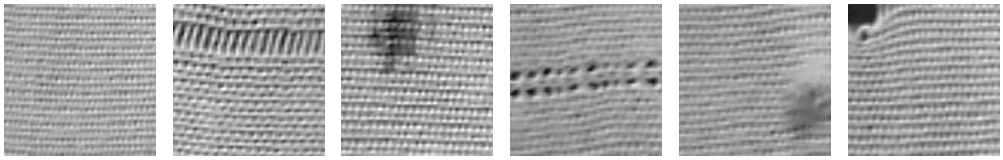
In this approach we defined six types of sample fabric classes, which were processed using images on the ARM processor. The imaging Factory library provided with the ARM processor was inappropriate for processing the images pixel by pixel. Therefore, we implemented our own graphic libraries for processing bitmap images.

Figure 5 shows examples of a non defective fabric sample together with samples of the five types of defects considered in this approach. The classes are differentiated using a Mahalanobis classifier, which uses the sample mean and the dispersion of the classes.[21] The classifier uses the Mahalanobis distance giben by:

$$d_{(i,j)} = (V_i - m_j)^T K_j^{-1} (V_i - m_j) \tag{7}$$

where m_j denotes the sample mean of Class j, V_i the input sample to the classifier, K_j^{-1} the covarianza matriz of the class j and $d_{(i,j)}$ is the distance between sample i and class j. The sample is associated to the class with minimal distance.

We use 50 images of class $C1$ and 10 images for each of the other classes to train the classifier. The images have been randomly chosen from a set of 1000 images of non defective samples and 2000 images of defective samples, all images with size of 100×100 pixels. We complement the images before applying the Haar wavelet decomposition. This is to obtain features with values around zero as an attempt to reduce the use of resources of the ARM processor by using small numbers in the majority of the calculations.



(a) C1 (b) C2 (c) C3 (d) C4 (e) C5 (f) C6

Figure 5: Types of images used in the database. a) non defective fabric. b) barre. c) stain. d) choppy. e) pill. f) hole.

We validate the classifier using 100 images. The results are shown in Table 1 in terms of correct classification using a confusion matrix. The confusion matrix is a squared table in which the results of assigned classes are shown on the columns while the actual classes are shown on the rows. Therefore, the sum of the columns in the table must be 100% but the sum of the rows may differ.

Table 1: Classification results within a confusion matrix

Actual Class	***Assigned class***					
	C1	***C2***	***C3***	***C4***	***C5***	***C6***
C1	98.0%	2.0%	0%	0%	0%	0%
C2	0%	96.7%	0%	0%	0%	3.3%
C3	0%	0%	85.0%	5.0%	5.0%	5.0%
C4	0%	14.3%	0%	71.4%	14.3%	0%
C5	0%	0%	0%	0%	100%	0%
C6	0%	0%	0%	0%	0%	100%

The results show that only 2% of non defective validated samples where incorrectly classified. The type of defect that was more incorrectly was the defect type choppy. We believe this is because the size of defect change from sample to sample and therefore we may include another constrain, such as size invariance, for its correct recognition.

5. CONCLUSIONES

Costs of automated inspection systems development can be reduced using modular solutions with embedded system, in which an important advantage is the low energy consumption. Portable devices with specific

tasks can be used in any on the inspection stages. Open architecture design, modularity and flexibility of embedded systems may permit the inspection to adapt to different stages within the manufacturing processes. Among the possibilities for developing embedded systems, we have explored the use of an ARM processor for defects detection by implementing the wavelet transform. The results suggest that accurate systems for specific applications can be developed with the ARM processors. However, a limitation that is still present is the speed of the processing, which is not yet sufficient for real time applications. Therefore, further research may be performed to optimize the functions to implement. Besides, a system combing ARM processors and parallel processors, such as GPU processors, may lead to develop a real time, modular, efficient and low cost systems accessible for small and medium companies.

REFERENCES

[1] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transaction on Pattern Analysis ang Machine Intelligence*, 11:674–693, 1989.

[2] G. Fan and X. Xia. Wavelet-based texture analysis and synthesis using hidden markov models. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 50:106–120, 2003.

[3] T. Chang and C. Kuo. Texture analysis and classification using tree-structured wavelet transform. *IEEE Transaction on Image Processing*, 2:207–248, 1993.

[4] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transaction on Image Processing*, 4:1549–1560, 1980.

[5] M. Weeks. *Digital Signal Processing Using MATLAB & Wavelets.* Jones and Bartlett Publishers, March 2010.

[6] A. Mojsilovic, M. V. Popovic, and D. M. Rackov. On the selection of an optimal wavelet basis for texture characterization. *IEEE Transactions on Image Processing*, 9(12):2043–2050, December 2000.

[7] Busch A and W. Boles. Texture classification using multiple wavelet analysis. In *Proceedings of the Digital Image Computing Techniques and Applications*, pages 1–5, 2002.

[8] Y. Han and P. Shi. An adaptive level-selecting wavelet transform for texture defect detection. *Image and Vision Computing*, 25(8):1239–1248, August 2007.

[9] E. P. Lam. Texture classification using wavelet decomposition. In *Proceedings of the IEEE International Conference on System of Systems Engineering*, pages 1–5, 2008.

[10] Z. Wang and J. Yong. Texture analysis and classification with linear regression model based on wavelet transform. *IEEE Transactions on Image Processing*, 17(8):1421?11430, August 2008.

[11] Y. Dong and J. Ma. Wavelet-based image texture classification using local energy histograms. *IEEE Signal Processing Letters*, 18(4):247–250, April 2011.

[12] K. Haapala, V. Lappalainen, and T. D. Hamalainen. Experimental parallel implementation of a wavelet-based still image encoder. *Microprocessors and Microsystems*, 29:155–167, 2004.

[13] J. Wang, M. Luo, J. Yang, F. Xu, H. Zhang, J. Zhao, G. Li, and K. Jiang. Real-time analysis system based on arm and dsp for audio-frequency stress wave. volume 9, pages V9–1 – V9–4. IEEE, 2009.

[14] J. P. Antoine. Image analysis with two-dimensionalcontinuouswavelettransform. *Signal Processing*, 31:241–272, 1993.

[15] J. Schmeelk. Wavelet transforms on two-dimensional images. *Mathematical and Computer Modelling*, 36:939–948, 2002.

[16] A. Fernández. *Estudio de técnicas basadas en la transformada wavelet y optimizacin de sus parámetros para la clasificación por texturas de imágenes digitales.* PhD thesis, Universidad Politecnica de Valencia, Febrary 2007.

[17] K. P. Soman and K. I. Ramachandran. *Insight Into Wavelets.* Motilal UK Books of India, October 2004.

[18] S. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, September 1999.

[19] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3:610–621, April 1973.

[20] R Conners and C. Harlow. Some theoretical considerations concerning texture analysis of radioghaphic images. *Sankhay*, 1980.

[21] P. Mahalanobis. Normalization of statistical variates and the use of rectangular co-ordinates in the theory of sampling distributions. *Sankhay*, pages 1–40, 1937.

[22] P. Mahalanobis. Handling borders in systolic architectures for the 1-d discrete wavelet transform for perfect reconstruction. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 48:1365–1378, May 2000.

[23] P. P. Ohanian and R. C. Dubes. Performance evaluation for four classes of textural features. *Pattern Recognition*, 25(8):819–833, August 1992.

[24] J. Strand and T. Taxt. Local frequency features for texture classification. *Pattern Recognition*, 27(10):1397–1406, October 1994.

[25] T. Randen and J. H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 21(4):291–310, April 1999.

[26] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, May 1979.